

Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets

Junyu Luo*

College of Computer Science
Sichuan University
Chengdu, P.R.China
2015141462158@stu.scu.edu.cn

March 30, 2017

Abstract

Generative Adversarial Net has shown its great ability in generating samples. The inverse mapping of generator also contains a great value. Some works have been developed to construct the inverse function of generator. However, the existing ways of training the inverse model of GANs have many shortcomings. In this paper, we propose a new approach of training the inverse model of generator by regarding a pre-trained generator as the decoder part of an autoencoder network. This model does not directly minimize the difference between original input and inverse output, but try to minimize the difference between the generated data by using original input and inverse output. This strategy overcome the difficulty in training a inverse model of a non one-to-one function. And the inverse mapping we learned can be directly used in image searching and processing.

1 Introduction

The paper of Deep convolutional generative adversarial nets [10] shows the great potential in the mapping between image space X and latent space Z . This relation can be used in the fields of image processing and searching. Finding the inverse mapping of generator can also provide us useful insights to the generator and we may use this to improve the performance of generator. However, the mapping from Z -space to image space X isn't a bijection. It brings a great challenge for finding the inverse mapping of generator.

Dumoulin [4] and Donahue [3] proposed a way of learning encoder network

*

alongside the generator and discriminator. This approach successfully avoid the problem of directly training the inverse model. However, this approach has a shortcoming that sample images and and corresponding reconstructions don't have a very good correlation. The main reason can be that the discriminator only focus on the difference between data sets instead of the difference between two images. So the encoder part can't catch the unique features of one signal image. In addition it needs to train a third network with the generative net, which means that inversion cannot be learned from a pre-trained generative network. Creswell and Bharath [2] proposed a different idea that can get a good correlation between samples and reconstructions. The main idea is to minimize the difference between generated image $G(z)$ and sample image x by change the value of z . This approach takes the desired output z as optimization goal. To get a corresponding z we have to use the gradient descent. It's very simple and easy. But it is not very practical because it doesn't provide a real inverse function. We have to calculate the z by using multiple gradient descents every time. In the invertible conditional GANs(ICGAN) proposed by Perarnau et al. [9], the inverse model are trained by directly minimizing the difference between $Encode(x)$ and (z, y) . $Encoder(x)$ has two outputs. The first is z and the second is label information vector y . The label information limits the freedom of z space. And the conditional vector y is helpful for training the inverse model. But this approach requires abundant label information and such data sets are very rare.

Hence we propose a new approach of learning the inverse mapping by AutoEncoder based on GANs(AEGAN) as a complement of their works. We use the idea of training the AutoEncoder to train a inverse model of a pre-trained generator. And this approach doesn't have the shortcomings mentioned above. The experiments show that AEGAN successfully learns the inverse mapping of original generator of GAN. And the corresponding vectors of images contains rich semantic information.

2 Related Work

2.1 Generative Adversarial Nets

A generative model aims to generate high quality artificial data by using adversarial strategy from game theory [5]. The GAN trains a discriminator and a generator at the same time and make them compete each other. The generator is aimed at approximating the underlying unknown data distribution to fool the discriminator, while the discriminator aims to tell which samples are real or fake. At the end the generator can learn the distribution of real data.

2.2 Autoencoder

AutoEncoder is an artificial neural network used for coding by unsupervised leaning [1]. The aim of an autoencoder is to learn the features of data and representing the data using the extracted features. In AutoEncoder, an encoder and a decoder are trained at a same time.. Encoder take original data as input and the decoder tries to reconstruct the original data using the output of encoder.

3 Autoencoder based Generative Adversarial Nets

The idea of AEGAN is inspired from Autoencoder. We define two models $G(z; \theta_g)$ and $D(x; \theta_d)$ representing the generator and discriminator of original GAN. And we define a inverse generator $IG(x; \theta_{ig})$ representing the inverse model of the generator. $\theta_g, \theta_d, \theta_{ig}$ are the parameters of the generator G , discriminator D and inverse generator IG . In here we regard the generator $G(z; \theta_g)$ as the decoder part of the Autoencoder and the desired inverse generator $IG(x; \theta_{ig})$ as the encoder part.

3.1 Training the Generator

First we train the generator using the normal approach of training GANs(The structure of GANs in this paper is based on the DCGAN of Radford [10]). The detail of training GANs can be found from the paper of Goodfellow [5].

The first training objective is:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))] \quad (1)$$

3.2 Training the Inverse Generator

Then we start to train the inverse generator IG by using the information of pre-trained generator G .

To avoid the difficulty of directly training the encoder we require the value function of IG to minimize the difference between generator's original output $G(z)$ and the reconstructed output $G(IG(G(z)))$. So we require z' can generate the same images as z instead of being same as z . This idea avoids the problem that mapping from Z to X is not a bijection.

In AEGAN, we use the cross-entropy cost function to represent the difference between $G(z)$ and the reconstructed output $G(IG(G(z)))$. The second training objective is:

$$\max_{IG} V(IG) = \mathbb{E}_{z \sim p(z)} [G(z) * \log G(IG(G(z))) + (1 - G(z)) * \log(1 - G(IG(G(z))))] \quad (2)$$

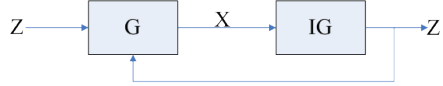


Figure 1: Structure of AEGAN

Figure 1 shows the basic structure of AEGAN. In AEGAN we don't directly minimize the difference between z and z' . We try to minimize the difference between $G(z)$ and $G(z')$. And Algorithm 1 shows the detail of training the inverse generator IG .

Algorithm 1 Training the Inverse Generator

for number of training iterations **do**
 Sample minibatch of m noise samples $(z^{(1)}, \dots, z^{(m)})$ from noise prior $z \sim p_g(z)$.
 Update the inverse generator by ascending its gradient:
 $\theta_{ig} = \theta_{ig} + \nabla_{\theta_{ig}} \frac{1}{m} \sum_{i=1}^m V(IG(z^{(i)}; \theta_{ig}))$
 where $V(IG(z^{(i)}; \theta_{ig}))$ is:
 $G(z^{(i)}) * \log G(IG(G(z^{(i)}))) + (1 - G(z^{(i)})) * \log(1 - G(IG(G(z^{(i)}))))$
end for

4 Experiment Results

We evaluate the ability of this inverse model on CelebFaces Attributes Dataset (CelebA) [8]. It's a large-scale face attributes dataset with more than 200K celebrity images. The all experiments below are down in unsupervised condition.

4.1 Reconstructing Samples

We take the outputs of generator as the samples and use the inverse mapping from these samples to Z space to generate the reconstructed samples. Here, we

compare AEGAN with a directly trained inverse model(This model is reformed from ICGAN [9]. We delete the conditional vector from the original model because of the unsupervised condition) and the adversarial based inverse model BiGAN of Donahue [3].

Figure 2 shows the reconstructed results of AEGAN and directly trained inverse model. Figure 3 shows the reconstructed results of BiGAN. Figure 3 use the different original samples because BiGAN can't use a pre-trained generator as base, so we compare BiGAN with the generated samples from its own generator for fairness.

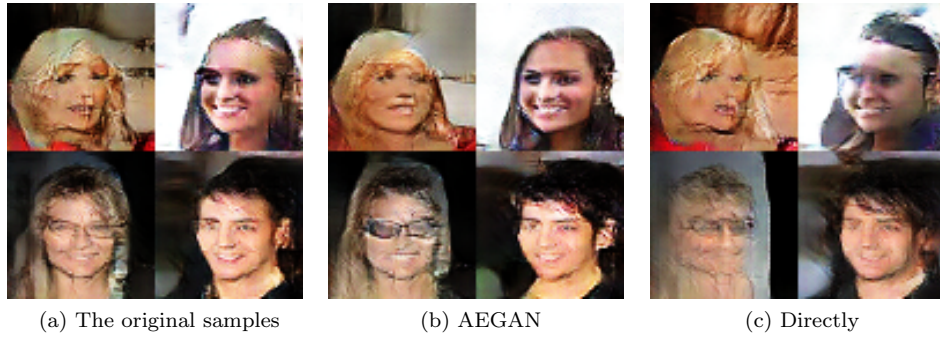


Figure 2: (a)the original samples, (b)the reconstructed samples using AEGAN, (c)the reconstructed samples using directly training

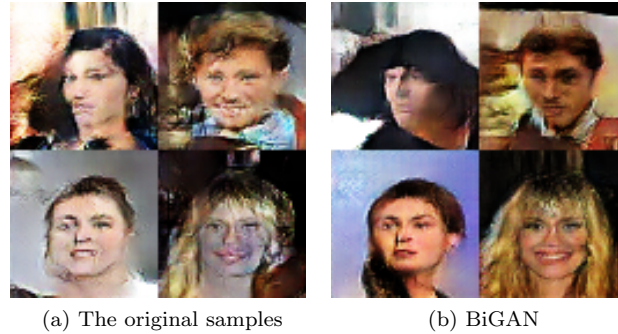


Figure 3: (a)the original samples, (b)the reconstructed samples of BiGAN

In addition, we use the dHash as standard to evaluate the similarity of generated images. We take the average similarity as final result. As we can see in Table 1, the result of AEGAN is also the best in this experimnt.

Table 1: Similarity compared with original samples

AEGAN	Directly training	BiGAN
0.8197	0.8069	0.6574

4.2 Searching the Similar Images Using AEGAN

To illustrate the value of AEGAN, we will show its ability in searching the similar images. In here, we only compare with the general image searching algorithm. Because our approach is based on unsupervised learning. We don't give out any additional information for this task. So it's fair to make our approach compete with them. We compare AEGAN with three general image searching algorithms: dHash, pHash and color histogram. In addition, we use Euclidean Distance of corresponding z vectors of images to get the similarity. If the distance is smaller we think two pictures are more similar.

We take a image from the original data set celebA and add some other factors to form the 3 test images. Then we implement 4 different algorithms to find the closest images in the

rst 20,000 images of celebA. The Figure 4 proves that AEGAN is very suitable for this task.

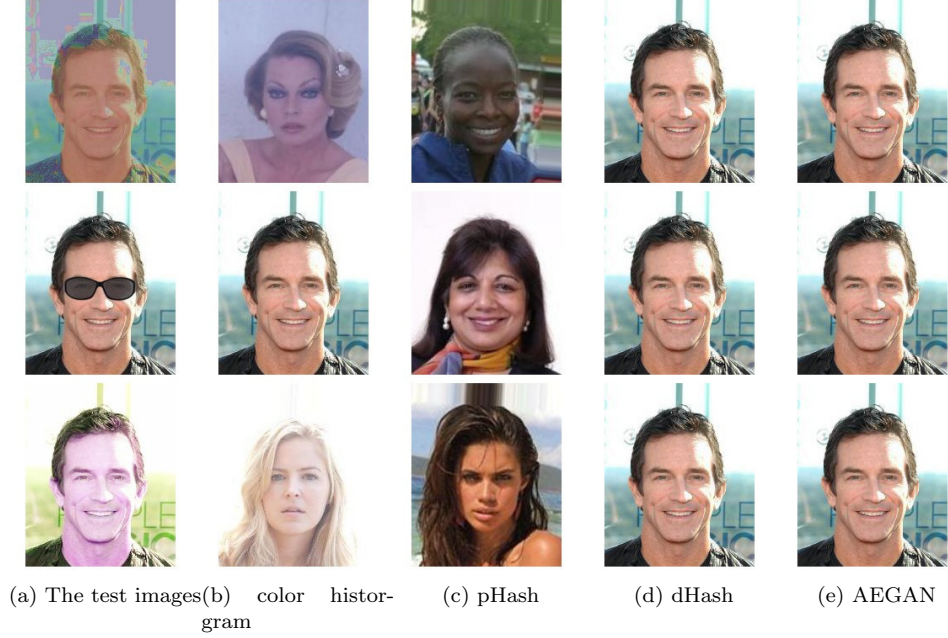


Figure 4: The searing results for color histogram, pHash, dHash and AEGAN.

To evaluate the comprehensive performance, the second experiment is aiming at finding the similar images. We only compare our algorithm with dHash, since it has a very good performance in the first experiment. We take the first 20,000 images of celebA as the test set. And take 64 images from the other part of the celebA as base images.

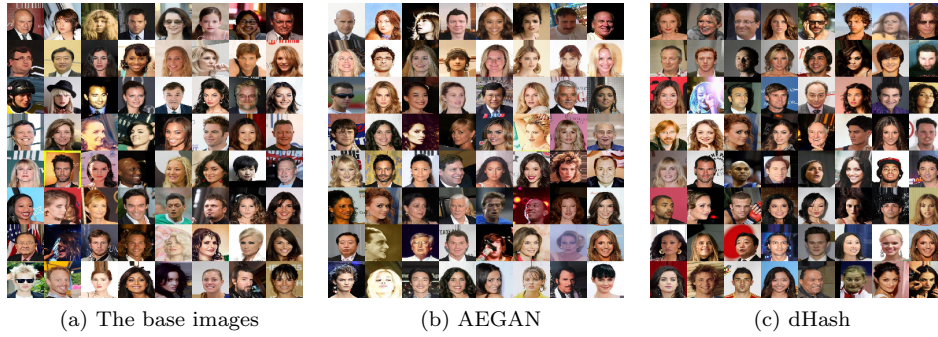


Figure 5: The searched results for dHash and AEGAN. (a) is the base images and (b), (c) are the searching results.

As we can see from Figure 5, AEGAN approach is much better than dHash. AEGAN catches the important features of face images such as the face angle, face similarity, hair style and expression. AEGAN may be not as good as specialized face recognition algorithms. But we have to emphasize again that this approach is unsupervised and universal. In other words, this idea can easily be implemented in other fields. We don't have to artificially find the best features for the task. This algorithm can automatically find the best features for the task and it only requires unlabeled data.

4.3 Super-Resolution Using the AEGAN

To prove our approach did learn the major features of face images, we propose the third experiment. In this experiment we take the Gaussian Blur images as inputs to inverse generator and then use the output of inverse generator to reconstruct the original images.

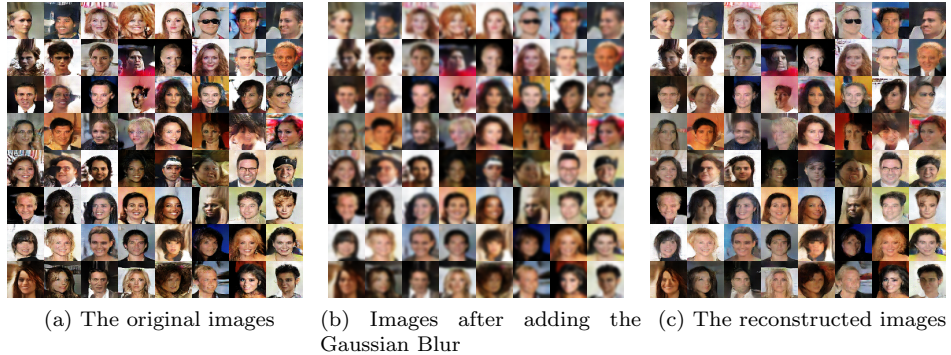


Figure 6: the results of super-resolution by AEGAN.

We choose the generated data as the original examples since the inverse model is aimed at learning the inverse mapping of generator. From Figure 6, we can see AEGAN also performs well in super-resolution.

5 Conclusion And Further Works

AEGAN uses the idea of Autoencoder to overcome the difficulty in training a inverse model of generator. And the experiments show that the inverse mapping of generator has a very similar function compared with Word Embedding [6]. This ability can be very helpful in fields of image and video processing. It's possible to get a universal vector representation of image if we train the AEGAN at large image data sets. And it's worth trying to apply this approach in video processing by first transforming the each frame of video to the corresponding image vector. In addition, we can use AEGAN to reform the Image-to-Image

Translation approach based on GAN [7]. With AEGAN, the training of generator part can be done in unsupervised condition and we only need to train the encoder part in conditional situation. In other words, it's possible to train a Image-to-Image GAN net in semi-supervised condition if we use the structure of AEGAN. However, one disadvantage of AEGAN is that the performance of AEGAN is limited by the power of generator. To get a good inverse mapping we need to train a good generator at first.

References

- [1] Yoshua Bengio. Learning deep architectures for ai. *Foundations & Trends in Machine Learning*, 2(1):1–55, 2009.
- [2] Antonia Creswell and Anil Anthony Bharath. Inverting the generator of a generative adversarial network. 2016.
- [3] Jeff Donahue, Philipp Krhenbhl, and Trevor Darrell. Adversarial feature learning. 2016.
- [4] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martin Arjovsky, Olivier Mastropietro, and Aaron Courville. Adversarially learned inference. 2016.
- [5] Ian J Goodfellow, Jean Pougetabadie, Mehdi Mirza, Bing Xu, David Wardefarley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, Zoubin Ghahramani, and Max Welling. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 3:2672–2680, 2014.
- [6] G E Hinton. Learning distributed representations of concepts. 1986.
- [7] Phillip Isola, Jun Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. 2016.
- [8] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. pages 3730–3738, 2014.
- [9] Guim Perarnau, Joost Van De Weijer, Bogdan Raducanu, and Jose M. Alvarez. Invertible conditional gans for image editing. 2016.
- [10] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *Computer Science*, 2015.